УДК 004.416.6

Ю.А. МАРИНЧЕНКО, А.Н. ЛИТВИНЕНКО

ОДНОРОДНЫЕ КОНСТРУКЦИИ

В данной статье вводится понятие однородных конструкций, которые реализуют принципы порождающего и расширяемого программирования. Однородные конструкции рассматриваются как конфигурационный ориентир программного проекта, определяющий направление процесса модуляризации и структурирования приложения. Описан ряд примеров однородных конструкций. Проанализировано место понятия однородных конструкций среди парадигм программирования, показаны достоинства использования однородных конструкций.

Ключевые слова: *однородные конструкции, инженерия предметной области, порождающее программирование, расширяемое программирование.*

Введение. Общепризнанно, что сопровождение и модернизация программных продуктов являются наиболее важными и ресурсоемкими этапами жизненного цикла приложения. Ключевым моментом для успешного сопровождения приложения является способ выделения частей программного кода, принципы и методы выделения модулей системы. Иными словами, это те конфигурационные ориентиры, которые используются при создании программного продукта. Так, выделение объектов в объектно-ориентированном программировании [2] – это один конфигурационный ориентир, выделение аспектов в аспектно-ориентированном программировании [1] – другой конфигурационный ориентир, вертикальные и горизонтальные слои Фуксмана [3] – третий конфигурационный ориентир. Однородные конструкции – это также один из конфигурационных ориентиров.

Постановка задачи. Требуется определить и разработать методику, которая, с одной стороны, позволяет организовывать модули приложения таким образом, что добавление новой функциональности или изменение существующей функциональности не затрагивает отлаженный программный код. Следовательно, применение подобной методики позволяет безболезненно изменять функциональность приложения с сохранением его работоспособности. С другой стороны, методика организации модулей представляет собой в простейшем случае некоторый параметризованный интерпретатор или параметризованный генератор исходного кода при более сложной реализации. Данная методика направлена на создание не конкретного программного средства, а серии настраиваемых программных средств. В данном случае разработка программного средства сводится к формированию высокоуровневой спецификации всего приложения или некоторых модулей приложения, по которой производится генерация программного кода приложения.

Методы испытаний. Для выполнения поставленной задачи использовались методы модульного и структурного анализа программных средств. Широкое применение получили методы инженерии предметной области, а именно: сбор, систематизация и сохранение наработанного опыта создания систем или частей систем в форме средств многократного применения. Ак-

тивно использовались методы программной инженерии – методы автоматизации жизненного цикла программного продукта, технологические методы разработки и тестирования сложных программных средств.

Понятие однородных конструкций. М.М. Горбунов-Посадов в своих трудах [6] выдвинул идею о прямой связи между расширяемостью и однородностью, т.е. в случае реализации однородного набора приложения специальным образом можно достичь расширяемости приложения, что, по его мнению, является ключом к развитию программы. Основываясь на сформулированных М.М. Горбуновым-Посадовым идеях [5], предложено точное определение и выработана структура однородных конструкций, представлены несколько примеров использования данной методики. Эти примеры протестированы и используются в приложениях, работающих в промышленном режиме.

Однородные конструкции являются способом организации модулей, реализующих некоторые однородные части программы. Данная методика реализует принципы порождающего [4] и расширяемого [5] программирования.

Под однородными конструкциями понимают набор однотипных элементов, имеющих одинаковые методы обработки. Элементы могут обладать достаточно сложной структурой. Однородная конструкция характеризуется репозиторием, процедурами обработки однородных конструкций и инструментом редактирования репозитория (рис.1).

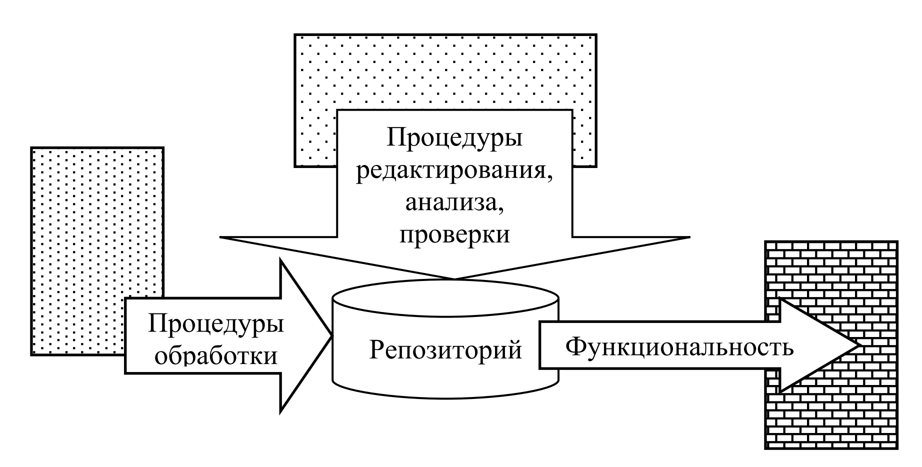


Рис.1. Структура однородных конструкций

Примеры однородных конструкций. Перед обзором примеров однородных пространств определим понятие "грид", которое является распространенным элементом любого приложения.

Грид (Grid) - это элемент пользовательского графического интерфейса, который отображает табличное представление данных и предоставляет следующую стандартную функциональность – редактирование данных в каждой ячейке, удаление и добавление строк.

Технология *PlugIn-модулей* является однородным пространством, где репозиторием однородного пространства служит определенная дирек-

тория, в которой расположены модули. Процедура обработки хранилища может запускаться при входе в приложение, по таймеру или в любой другой точке приложения. После запуска процедуры обработки репозитория появляется функциональность PlugIn-модулей в соответствующих точках приложения. Для добавления новой функциональности в виде PlugIn-модуля достаточно просто поместить файл определенной структуры в PlugIn-директорию. Для устранения функциональности модуля достаточно убрать PlugIn-модуль из директории.

Отчетные формы. Все отчеты приложения удобно рассматривать как однородное пространство, в котором хранилище представлено dbf-файлом. При удалении какого-либо элемента однородного пространства из хранилища вся функциональность, соответствующая данному элементу, автоматически исчезает из приложения. В частности, если нужно изменить функциональность отчета, то редактирование затрагивает только файл исходного кода, содержащий процедуру формирования отчета, при этом ядро приложения остается неизменным. Добавление, удаление элементов однородного пространства можно реализовать, используя технологию PlugIn-модулей, где один отчет является PlugIn-модулем, который содержит процедуры формирования отчета, а также процедуру регистрации последнего в системе. Регистрация PlugIn-отчета заключается в добавлении или изменении соответствующего элемента в репозитории отчетов. Отчет может характеризоваться не только процедурами формирования отчета, но и шаблоном отчета, представленном, например, в формате rpt, rtf или xls. Для того чтобы добавить новый отчет в приложение, нужно занести всю необходимую информацию в репозиторий отчетов. Добавление нового элемента однородного пространства в репозиторий отчетов с использованием PlugIn-технологии сводится к размещению файлов шаблона и процедур регистрации и формирования отчета в соответствующие директории. Добавление нового элемента однородного пространства без использования технологии PlugIn-модулей осуществляется добавлением нового элемента непосредственно в хранилище отчетов, при это нужно помнить и о возможном шаблоне отчета, который следует поместить в соответствующую директорию.

Типовые операции для грида. Рассмотрим однородное пространство типовых операций для грида. Типовые операции подразделяются на стандартные (количество строк, сумма по столбцу, поиск значения и т.д.) и нестандартные, которые применимы для конкретных гридов, например, для конкретной строки грида "Заказы" вывести на экран форму заказа. Добавление, удаление типовых операций производится путем вставки, удаления элемента однородного пространства из репозитория. Модификация функциональности любого элемента осуществляется также в хранилище и не затрагивает исходный код ядра приложения. Каждая типовая операция характеризуется условием допустимости, при помощи которого можно указать грид или набор гридов, где используется функциональность типовой операции, или наоборот, определить грид или набор гридов, для которых типовая операция не действует.

В итоге, разработав один раз процедуры обработки репозитория типовых операция для грида, последующее добавление и удаление операций сводится к редактированию хранилища однородного пространства. Среди процедур обработки репозитория выделим процедуру выбора допустимых операций для грида в контекстное меню, процедуру запуска конкретной операции. Реализация типовых операций без использования однородных пространств заключается в определении контекстного меню для каждого грида приложения. А количество гридов в серьезном приложении очень велико, более того, контекстные меню различных гридов не совпадают. При такой реализации значительно затрудняется процесс сопровождения. Предположим, что необходимо добавить новую операцию во все гриды приложения. В случае однородных пространств это делается элементарно: добавляем в репозиторий пространства новый элемент с условием допустимости "ИСТИНА". В реализации без использования однородных пространств нужно внести изменения в контекстное меню всех гридов, при этом ни один не забыть. Использование последнего способа реализации типовых операций повышает вероятность появления ошибки, значительно затрудняет процесс внесения изменений. Изменения рассредоточены по исходному коду приложения и носят один и тот же характер.

Состав однородных конструкций. Следует обратить особое внимание на то, что однородные конструкции, принадлежащие одному набору, должны иметь одинаковую структуру, которая определяет *репозиторий рассматриваемого набора однородных конструкций.* Следовательно, важным этапом реализации наборов однородных конструкций является определение структуры однородных пространств, а именно определение типа и атрибутов хранилищ однородных конструкций в зависимости от рода решаемых задач и возможностей среды разработки. Хранилище является обязательным составляющим однородных конструкций.

Выделим следующие способы организации репозитория однородных конструкций: текстовый файл; файл разметки XML; DBF-файл. Каждый из способов организации репозитория однородных конструкций имеет свои преимущества и недостатки.

XML-файл более предпочтителен в качестве репозитория однородных конструкций в силу того, что хранилище подобного типа может вместить в себя данные со сколь угодно сложной структурой (рис.2). В XML-репозитории не существует ограничений на типы атрибутов в отличие от DBF-хранилища. Например, изображенный ниже элемент однородной конструкции достаточно трудно было бы реализовать в DBF-файле, так как типы атрибутов DBF-хранилища должны быть простыми.

```
<reports>
  <repgroup id="7" cname="Продажи">
          <entry rep1_id="1" repname="Продажа" sit="rep" ord="2.0000">
                          <cprg>DO repprdg WITH [r] IN _repprdg</cprg>
                          <uslov/>
                          <prim/>
          </entry>
  </repgroup>
</reports>
```

Рис.2. Пример элемента однородной конструкции сложной структуры

Ещё одним преимуществом использования XML-файлов является то, что для редактирования этого файла не требуется никакого дополнительного инструментария, кроме обычного редактора текстовых файлов. Для изменения элементов DBF-хранилища необходимо наличие некоторого программного обеспечения, позволяющее удалять, добавлять, модифицировать DBF-файл. Предоставляется поддержка проверки корректности XML-файла путем использования XSD-схем.

Как уже отмечалось ранее, для каждого набора однородных конструкций необходимо разработать универсальные *процедуры обработки элементов однородного пространства*, которые, по определению, должны быть инвариантны относительно мощности однородного пространства.

Предполагается, что процедуры обработки набора однородных конструкций реализуют функцию интеграции ядра приложения с элементами однородного пространства, т.е. выступают некоторым параметризованным интерпретатором или генератором частей исходного кода [4] в сложном случае. Параметры берутся из репозитория однородных конструкций.

Следовательно, ядро приложения и однородные конструкции находятся в отношении слабосвязанности и взаимозависимости одновременно. Проявление взаимозависимости очевидно, так как ядро приложения предоставляет пользователю функциональность однородных пространств, процедуры однородного пространства, в свою очередь, считывают данные объектов приложения, используют параметры, переданные приложением на вход процедурам однородного пространства. Слабосвязанность характеризует способ взаимосвязи между ядром приложения и однородным пространством, метод переплетения ядра и процедур однородных конструкций. Слабосвязанность можно определить как характеристику интеграции модулей приложения. Подобная интеграция позволяет безболезненно вносить изменения в приложения таким образом, что модификации одного модуля не влияют на работоспособность другого модуля в силу высокой степени инкапсуляции функциональности модулей. Стандартным примером реализации отношения слабосвязанности и взаимозависимости являются PlugIn-модули, описанные выше.

Инструмент редактирования репозитория однородных конструкций является опциональным составляющим однородных конструкций. Данный инструментарий должен быть наделен функцией редактирования, анализа и проверки корректности элементов однородных конструкций при добавлении и изменении хранилища. Набор средств для редактирования, анализа и проверки корректности однородных конструкций может быть реализован программистом, а может входить в состав некоторых типов репозитория. В частности, в случае использования dbf-хранилища проверка корректности производится в соответствии со структурой и ограничениями данного типа хранилища независимо от наличия инструментария редактирования и анализа, созданного разработчиком, но такой тип хранилища и не исключает последнего.

Набор средств для редактирования и анализа репозитория может вызываться при разработке приложения (designtime), в любой точке в процессе работы приложения (runtime). Следует обратить особое внимание на

вызов инструмента редактирования и анализа репозитория при входе в программу. Данная точка вызова относится к точкам вызова в режиме runtime, но именно она наиболее часто используется в приложениях. Встречаются реализации набора средств для редактирования и анализа репозитория, когда инструментарий редактирования является частью приложения и доступен пользователю в любой момент времени работы приложения.

Место однородных конструкций в теории программирования. Проблематику аккумулирования опыта создания систем в некоторой предметной области, а также дальнейшего использования накопленного опыта рассматривает инженерия предметной области [4]. Инженерия предметной области (Domain Engineering) — это деятельность по сбору, систематизации и сохранению наработанного опыта создания систем или частей систем в форме средств многократного применения, т.е. повторно используемых рабочих средств в рамках определенной предметной области, а также по обеспечению методов для повторного использования этих средств, т.е. поиска, классификации, распространения, адаптации, сборки и т.д. в процессе создания новых систем.

Назначение методики однородных пространств является аккумулирование опыта разработки приложений, создание и использование при дальнейшей работе определенных элементов конфигурации приложения, а именно, конкретных однородных пространств. Следовательно, методика однородных пространств реализует принципы инженерии предметной области, где в роли предметной области выступает конфигурация приложения. Применение принципов инженерии предметной области качественно влияет на процессы программной инженерии, а именно, процесс разработки и сопровождения приложения.

Принципы инженерии предметной области активно используются в порождающем [4] и расширяемом [5] программировании. Основной упор в парадигме порождающего программирования [4] делается на приемах разработки серии настраиваемых приложений, а именно, на приемах генерации или интерпретации приложения из уже разработанных универсальных многократно используемых модулей, содержащих определенную функциональность, путем указания конфигурации приложения или конфигурации некоторой части приложения. Именно в этих универсальных модулях происходит накопление и использование аккумулированного опыта в конкретной предметной области.

Следует отметить, что изменения по характеру можно разделить на революционные, которые в корне меняют структуру проекта, и эволюционные, которые преобразуют или дополняют только некоторую часть проекта. Реализация эволюционных изменений составляет основную долю работы персонала по сопровождению приложения, т.е. эволюционные изменения являются точками роста программы.

Основной идеей расширяемого программирования [5] является идея организации и модуляризации приложения таким образом, чтобы дальнейшее развитие комплекса не требовало редактирования уже написанного программного кода. Вновь написанные части программы будут только дополнять исходные тексты приложения. Безболезненное сопровождение системы теоретически достигается путем использования следующих составляющих конфигурации проекта: каркасы, вариантные гнезда, наборные гнезда и т.д. Согласно М.М. Горбунову-Посадову [5] безболезненность изменений приложения достигается тем, что изменения не затра-

гивают уже отлаженный программный код приложения, а следовательно, не влияют на работоспособность приложения.

С одной стороны, методика однородных конструкций является прямым воплощением принципов расширяемого программирования [5], так как направлена на упрощение процесса сопровождения, когда добавление, изменение или устранение функциональности не требует редактирования исходного кода ядра приложения. В данной методике используется теория каркасов и гнезд.

С другой стороны, методика однородных пространств реализует принципы порождающего программирования [4] потому, что процедуры обработки репозитория однородных конструкций выступают в роли параметризованных интерпретаторов или генераторов фрагментов исходного кода приложения в более сложных случаях.

Особенности реализации однородных конструкций. Основная сложность реализации однородных конструкций заключается в разработке универсальнальной процедуры обработки конструкций, а именно, в выработке техники генерации исходного кода в соответствии с элементами однородного пространства. К этой задаче необходимо подойти очень серьезно, так как от эффективности использования данной техники зависит работа и дальнейшее развитие и работоспособность всего приложения.

Все описанные сложности связаны с тем, что реализация однородных конструкций является решением более общей задачи для программиста, чем разработка конкретной функциональности. Поэтому требуется затратить больше усилий на анализ и разработку хранилища и универсальной процедуры обработки элементов хранилища, которые бы позволили разместить любой набор элементов предметной области данного однородного пространства. Однако все затраченные усилия на разработку однородных конструкций в дальнейшем с лихвой окупаются.

Однородные конструкции являются воплощением основных принципов расширяемого [5] и порождающего [4] программирования, поскольку добавление очередного элемента однородного пространства сводится к занесению новой позиции в репозиторий однородного пространства, что не сопровождается изменением исполняемого кода приложения. Следовательно, в терминах порождающего программирования генератором некоторого программного кода приложения является модуль универсальной обработки однородных конструкций, а высокоуровневой спецификацией, по которой происходит генерация кода, служит хранилище однородных конструкций.

Модуль - выделенная по тем или иными мотивам часть первичного материала программного фонда [5]. Таким образом, учитывая принципы расширяемого программирования, определен новый конфигурационный ориентир [5], которым является однородное пространство. Использование однородной конструкции в качестве способа модуляризации программных средств позволяет свести процесс разработки к созданию хранилищ однородных пространств и процедур обработки последних. Далее строятся элементы однородного пространства, разрабатывается функциональность каждого элемента, что не влияет на работоспособность ядра приложения. Подобная модуляризация позволяет безболезненно дополнять и изменять функциональность приложения, легко локализовать ошибочный код после внесения изменений.

Примеры реализации однородных конструкций. Большинство приложений работают с данными независимо от специфики последнего, следовательно, можно выделить такие неотъемлемые задачи приложения, как

запрос и фильтрация данных, организация отчетности и выполнение некоторых типовых операций для грида. В связи с этим было выделено несколько классов однородных конструкций, которые подробно будут описаны далее.

Крайне полезно организовать *набор операций* для грида, а также предусмотреть возможность *добавления операций*, доступных при выполнении некоторых условий. Следовательно, можно определить однородное пространство наборов операций для грида, где элементом пространства является операция.

Для данного однородного пространства создан xml-репозиторий элементов пространства, имеющий следующие атрибуты: уникальный идентификатор, наименование, условие допустимости, строку запуска и приоритет операции (рис.3).

```
<myoper>
   <oper id="rep_ds" name="Отчет 'Продажи'" prior="1.00000">
                   <do>do repprdg with 'ds' in _repprdg</do>
                   <uslov>USED('rv_uz') AND USED('rv_uz1')</uslov>
                   <prim>D</prim>
   </oper>
</myoper>
```

Рис.3. Элемент однородного пространства операций для грида

С точки зрения приложения предоставлены следующие действия с типовыми операциями для грида: просмотр списка допустимых операций; выполнение конкретной операции грида.

Функция просмотра списка допустимых операций инициируется нажатием на правую кнопку мыши на гриде, тогда осуществляется поиск допустимых операций в репозитории типовых операций в соответствии с условием допустимости. Условие допустимости определяется логическим выражением, результат которого меняется в зависимости от переменных среды окружения в некоторой точке приложения. При этом происходит формирование контекстного меню, содержащего допустимые операции однородного пространства в соответствии с условием допустимости.

Функция выполнения конкретной операции по репозиторию определяет строку запуска данной операции и производит её исполнение.

Фильтрация записей – стандартная и распространенная операция, позволяющая пользователю выбирать нужные ему данные из всего объема данных. Пожалуй, нет таких приложений, в которых бы не использовались различные модули фильтрации. Для построения условия фильтрации необходимо указать поле грида, которое участвует в отборе, непосредственно условие фильтрации ("=","<",">","В СПИСКЕ","ПУСТО"), а также значение условия отбора, которое можно выбрать из списка представленных в гриде значений или задать независимо от списка значений.

Разработанный фильтр учитывает не только простые условия, налагаемые на поля грида, но также и достаточно сложные и нетривиальные условия фильтрации, использующие данные дочерних гридов.

Например, в приложении представлен грид "Заказы" и "Заказчики". Грид "Заказы" ссылается на грид "Заказчики". Предположим, что необходи-

мо построить условие фильтрации, выбирающее заказы, имеющие в качестве заказчика индивидуального предпринимателя. Такое условие невозможно реализовать, используя простые условия фильтрации, поэтому сложные условия фильтрации рассмотрим отдельно.

Наборы упомянутых сложных условий фильтрации выбраны в качестве однородного пространства. Элементом однородного пространства является условие фильтрации или запроса, которое имеет следующие атрибуты: наименование, выражение допустимости, список возможных операций, применимых к условию, и соответствующие команды выполнения условия фильтрации. В качестве репозитория однородного пространства выбран xml-файл с достаточно простой разметкой, учитывающей все перечисленные атрибуты, но в данном случае структура хранилища не является плоской, так как некоторые атрибуты имеют достаточно сложное строение. Редактирование хранилища производится путем редактирования xml-файла.

Однородное пространство условий фильтрации/запроса предполагает определение следующих функций: просмотр допустимых условий фильтрации/запроса в данной точке приложения; просмотр возможных операций, применимых для конкретного условия; просмотр списка возможных значений для выбранного условия.

Функция просмотра допустимых условий фильтрации или запроса вызывается только программно модулем фильтрации или запроса данных при построении нового условия фильтрации. Данная функция производит поиск по репозиторию и выбор допустимых условий фильтрации в данной точке приложения, основываясь на выражении допустимости, которое принимает логическое значение в зависимости от переменных среды окружения. Зачастую выражение допустимости содержит проверку на принадлежность условия фильтрации экранной форме, гриду или некоторому набору гридов. Результатом выполнения функции просмотра допустимых условий фильтрации или запроса является список наименований условий, применимых в данной точке приложения.

Для одного условия фильтрации может быть определено несколько операций. Операции представляют собой как простые операции сравнения, так и более сложные операции, например, определение списка значений данного поля (Тип товара В СПИСКЕ ('Обувь','Стройматериалы','Оргтехника')). Функция просмотра возможных операций, применимых для условия фильтрации, осуществляет поиск по хранилищу и выбор определенных для данного условия операций. Функция просмотра списка возможных значений для конкретного условия фильтрации возвращает список результатов выполнения выбранной из репозитория команды.

Одной из важных задач приложения является *организация отчетности* различного рода. Разработка стандартного механизма для запуска и вывода отчета значительно повысит эффективность проектирования и сопровождения приложения.

Рассмотрим однородное пространство отчетов, где элементом пространства является отчет. Репозиторием однородного пространства выбран dbf-файл, содержащий информацию о следующих атрибутах: уникальный идентификатор, наименование, строка запуска процедуры предварительной обработки, строка запуска отчета, а также условие допустимости.

С точки зрения приложения возможны следующие операции с отчетами: просмотр допустимых отчетов в текущей точке приложения, а также запуск конкретного отчета.

Просмотр списка допустимых отчетов может инициироваться выбором пункта меню, иконки на панели инструментов, нажатием на кнопку экранной формы или некоторой комбинацией "горячих" клавиш. Независимо от способа инициации выбора отчетов производится поиск отчетов в соответствии с условием допустимости, которое указано в репозитории однородного пространства и представляет собой логическое выражение. Результатом выполнения условия допустимости является истина или ложь в зависимости от значений переменных среды окружения в конкретной точке приложения.

Процедура запуска конкретного отчета в качестве входных параметров получает уникальный идентификатор отчета из хранилища однородных конструкций и способ вывода данных отчета, среди которых выделены вывод на принтер, в файл и на экран. По уникальному идентификатору происходит поиск отчета в репозитории, затем определяется необходимость запуска процедуры запроса данных для отчета. Потом происходит формирование и вывод выбранного отчета с учетом входных параметров и данных репозитория. В большинстве случаев для формирования отчета требуется шаблон, расположение которого также указывается в хранилище отчетов. Шаблоны могут быть разных типов, начиная с frx-отчета и заканчивая xls-шаблоном для вывода в Microsoft Office Excel.

Выводы. Понятие однородных конструкций имеет большое значение в процессе структурирования приложения. Применение данного понятия при разработке программного средства дает ряд преимуществ. Основным достоинством является простота добавления новой и изменения существующей функциональности, легкость локализации и устранения ошибок редактирования программного кода.

Наличие стандартной процедуры обработки однородных конструкций позволяет вносить изменения в функционал без модификации ядра приложения, что указывает на высокую степень открытости данного приложения.

Применение однородных конструкций воплощает в жизнь основные принципы инженерии предметной области [4]. В данных конкретных реализациях, рассмотренных выше, аккумулирование и использование опыта разработки приложение инвариантно относительно языка программирования, так как представленные типы хранилищ однородных конструкций (XML-файл, текстовый файл, DBF-файл) легко распознаются любым современным языком программирования. Конечно, для использования готовых однородных конструкций в любой другой среде разработки недостаточно просто хранилища, нужно будет адаптировать существующую или реализовать новую процедуру обработки элементов однородного пространства.

Использование однородных конструкций позволит значительно оптимизировать сопровождение приложения и упростит его модификацию.

Библиографический список

1. Kiczales G., Lamping J., Mendhekar A. Aspect-oriented programming // http://www.oberon2005.ru/paper/gk1997.pdf.
2. Фридман А.Л. Основы объектно-ориентированной разработки программных систем /А.Л.Фридман. – М.: Финансы и статистика, 2000. – С.192.
3. Фуксман А.Л. Технологические аспекты создания программных систем /А.Л.Фуксман. – М.: Статистика, 1979. – С.184.
4. Чарнецки К. Порождающее программирование: методы, инструменты, применение /К.Чарнецки, У.Айзенекер. – СПб: Питер, 2005. – С.736.
5. Горбунов-Посадов М.М. Расширяемые программы /М.М.Горбунов-Посадов. – М.: Полиптих, 1999. – С.336.
6. Горбунов-Посадов М.М. Как растет программа // Препринт Института прикладной математики им. Келдыша М.В. РАН. – 2000. – №50. – С.16.
7. Липаев В.В. Программная инженерия в жизненном цикле программных средств //http://www.citforum.ru/SE/lipaev.

Материал поступил в редакцию 15.10.08.

Y.A. MARINCHENKO, A.N. LITVINENKO

HOMOGENEOUS STRUCTURES

The concept of homogeneous structures which realize principles of generating and expanded programming is introduced. Homogeneous structures are considered as the configuration reference point of the program project defining a process direction of division into modules and program structurization. Same examples of homogeneous structures are described. The place of concept of homogeneous structures among programming paradigms is analysed. Advantages of the homogeneous structures's use are shown.

МАРИНЧЕНКО Юлия Александровна, аспирант ЮФУ. Окончила Ростовский государственный университет (2004).
Область научных интересов: развитие, доработка, расширение возможностей программного обеспечения. Совершенствование программного обеспечения
Количество публикаций: 12.

ЛИТВИНЕНКО Александр Николаевич (р.1956), кандидат технических наук (1990), доцент кафедры ИиВЭ ЮФУ. Окончил Ростовский государственный университет (1978).
Область научных интересов: программная инженерия, базы данных
Количество публикаций: 35.